

Docs as Code. Базовые понятия

Что такое Docs as Code

Docs as code – подход к разработке документации, основанный на принципах и практиках работы с кодом.

Цель подхода Docs as Code

Создание, развитие и поддержка актуальной, масштабируемой, многопользовательской системы документации и прозрачных процессов вокруг неё.

Унификация текста

1. Использование стайлгайдов, ред.политики, etc.
2. Использование шаблонов
3. Переиспользование контента – единый источник
4. Унификация верстки текста – языки разметки

Языки разметки

Язык разметки – это язык, предназначенный для форматирования и структурирования текста с помощью комбинаций разных символов (разметки), которые добавляются в текст и сообщают, как именно он должен быть организован.

Цель языка разметки – отделить содержание от формы.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Настройка файрвола в RedHat Linux</title>
</head>
<body>

<h1>Настройка файрвола в RedHat Linux</h1>

<p>Чтобы настроить файрвол в RedHat Linux, воспользуйтесь утилитой <code>iptables</code>. </p>

<p>Создайте набор правил, по которым будет контролироваться входящий, исходящий и проходящий трафики сетевых пакетов. Правила входят в цепочки:</p>
<ul>
    <li><code>INPUT</code> (все входящие пакеты);</li>
    <li><code>OUTPUT</code> (все исходящие пакеты);</li>
    <li><code>FORWARD</code> (все проходящие пакеты, не предназначенные этому серверу).</li>
</ul>

<p>Для каждого правила определяется действие:</p>
<ul>
    <li><code>ACCEPT</code> – разрешить пакет;</li>
    <li><code>DROP</code> – выбросить пакет без уведомления отправителя;</li>
    <li><code>REJECT</code> – отказать в прохождении пакета с уведомлением отправителя.</li>
</ul>

<ol>
    <li>
        <p>Перед созданием набора правил определите политики цепочек по умолчанию.<br>
        Выполните последовательно команды:</p>
        <pre><code>iptables -p INPUT ACCEPT
iptables -p OUTPUT ACCEPT
iptables -p FORWARD ACCEPT
</code></pre>
        <p>Пакеты, которые не попадают ни под одно из правил, по умолчанию будут пропускаться файрволом.</p>
    </li>
    <li>
        <p>Создайте правило, которое разрешает уже установленные соединения:</p>
        <pre><code>iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
</code></pre>
        <p><code>-A</code> обозначает добавление нового правила; <br>
        <code>-j</code> – ключ для указания действия.</p>
    </li>
</ol>

```

Облегчённые языки разметки

Облегчённый язык разметки – это язык разметки с простым синтаксисом, который удобно редактировать и читать в исходном виде: markdown, reStructuredText (RST), AsciiDoc

Настройка файрвола в RedHat Linux

Чтобы настроить файрвол в RedHat Linux, воспользуйтесь утилитой `iptables`.

Создайте набор правил, по которым будет контролироваться входящий, исходящий и проходящий трафики сетевых пакетов. Правила входят в цепочки:

- INPUT (все входящие пакеты);
- OUTPUT (все исходящие пакеты);
- FORWARD (все проходящие пакеты, не предназначенные этому серверу).

Для каждого правила определяется действие:

- ACCEPT – разрешить пакет;
- DROP – выбросить пакет без уведомления отправителя;
- REJECT – отказать в прохождении пакета с уведомлением отправителя.

1. Перед созданием набора правил определите политики цепочек по умолчанию.

Выполните последовательно команды:

```
...
iptables -p INPUT ACCEPT
iptables -p OUTPUT ACCEPT
iptables -p FORWARD ACCEPT
...
```

Пакеты, которые не попадают ни под одно из правил, по умолчанию будут пропускаться файрволом.

2. Создайте правило, которое разрешает уже установленные соединения:

```
...
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
...
```

`-A` обозначает добавление нового правила;

`-j` – ключ для указания действия.

3. Создайте правила, которые разрешают icmp-пакеты с типами ICMP сообщений `Echo` и `Time exceeded`:

```
...
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
...
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
...
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Настройка файрвола в RedHat Linux</title>
</head>
<body>
```

Настройка файрвола в RedHat Linux

Чтобы настроить файрвол в RedHat Linux, воспользуйтесь утилитой `iptables`.

Создайте набор правил, по которым будет контролироваться входящий, исходящий и проходящий трафики сетевых пакетов. Правила входят в цепочки:

- `INPUT` (все входящие пакеты);
- `OUTPUT` (все исходящие пакеты);
- `FORWARD` (все проходящие пакеты, не предназначенные этому серверу).

Для каждого правила определяется действие:

- `ACCEPT` – разрешить пакет;
- `DROP` – выбросить пакет без уведомления отправителя;
- `REJECT` – отказать в прохождении пакета с уведомлением отправителя.

Перед созданием набора правил определите политики цепочек по умолчанию.

Выполните последовательно команды:

```
iptables -p INPUT ACCEPT
iptables -p OUTPUT ACCEPT
iptables -p FORWARD ACCEPT
...
```

Пакеты, которые не попадают ни под одно из правил, по умолчанию будут пропускаться файрволом.

- `-A` обозначает добавление нового правила;
- `-j` – ключ для указания действия.

Версионирование и командная работа

Система контроля версий – программное обеспечение для работы с регулярно обновляющимися данными, которое позволяет хранить несколько версий одного и того же файла и отслеживать его изменения.

- Локальные
- Централизованные
- Распределённые

GIT – распределённая система контроля версий

GitHub, GitLab, GitVerse – web-сервисы для хранения и управления репозиториями

Ревью документации

- Формализация процесса вычитки текстов
- Организация кросс-ревью
- Использование функциональности GitHub (комментирование в PR, апрувы).

Сборка, тестирование и публикация документации. CI/CD

Генератор статических сайтов – инструмент, который автоматически конвертирует исходные текстовые файлы в html-страницы.

CI/CD – набор практик и инструментов для автоматического тестирования и развёртывания приложений.

CI (continuous integration) – непрерывная интеграция

Тестирование документации

CD (continuous delivery/deployment) – непрерывная доставка/развёртывание

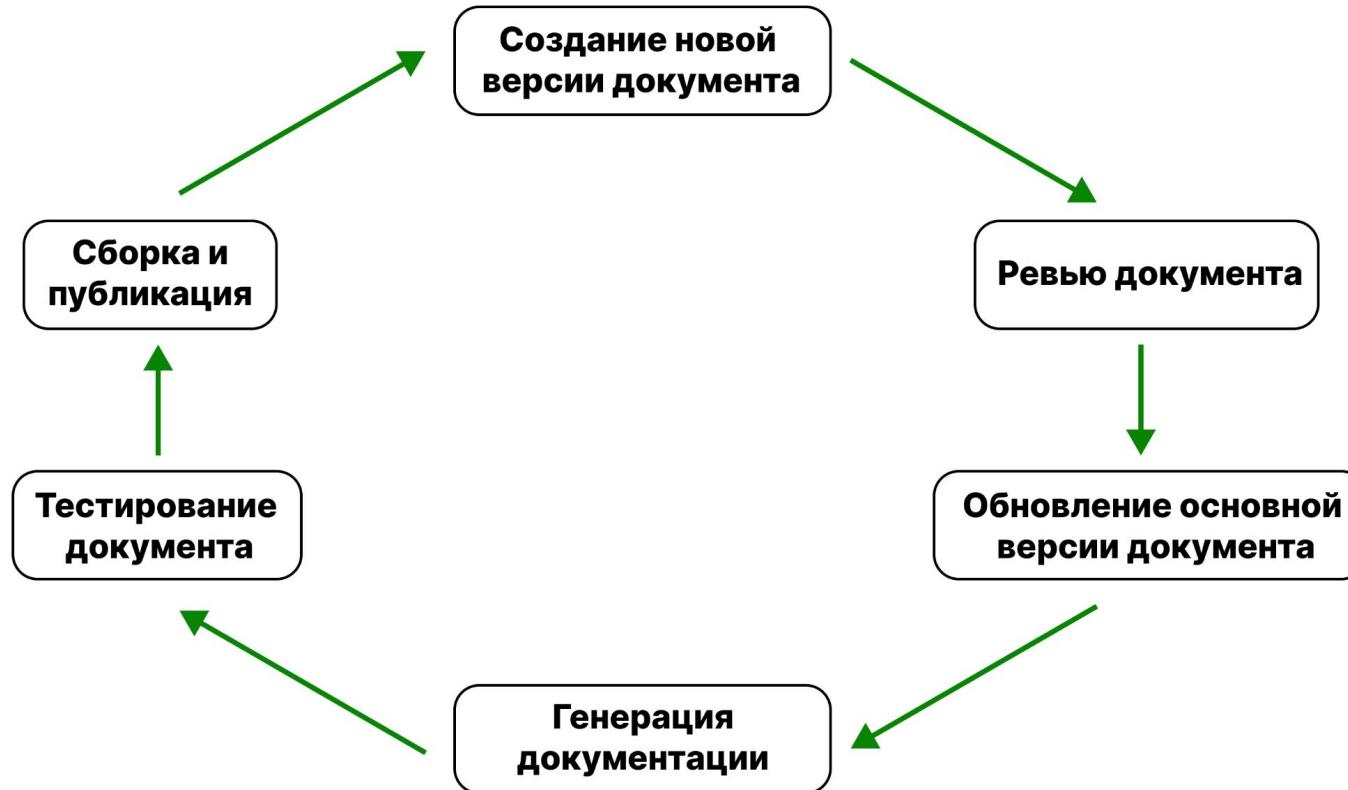
Публикация документации

Жизненный цикл* документа

— — —

Путь документа от постановки задачи до публикации





Преимущества

- Хранение документации рядом с кодом
- Интеграция релизов документации с релизами продукта
- Совместная работа с документацией
- Прозрачный процесс документирования

О большом мастер-классе “Docs as Code advanced”

- GIT для технического писателя
 - Практическая работа с git
- Ревью текста. Практическая работа
- Разворачивание проекта MkDocs Material
 - Практическая работа по настройке проекта
- Публикация проекта. Автоматизация сборки
 - Практическая работа
- Кастомизация проекта. Авторские стили
 - Практическая работа
- Формирование портфолио. Диаграмма последовательности и спецификация OpenAPI
 - Практическая работа

Домашнее задание №1

1. Познакомьтесь с базовым синтаксисом Markdown:
<https://novillero.github.io/dac-advanced/markdown/>
2. Напишите пошаговую инструкцию для любого пользовательского сценария в телеграм и оформите её с помощью разметки markdown.